A PRE- AND POST-PROCESSOR FOR THE ICOOL MUON TRANSPORT CODE *

W.M. Fawley†, LBNL, Berkeley, CA 94720, USA

Abstract

ICOOL[1] is a Fortran77 macroparticle transport code widely used by researchers to study the front end of a neutrino factory/muon collider[2]. In part due to the desire that ICOOL be usable over multiple computer platforms and operating systems, the code uses simple text files for input/output services. This choice together with user-driven requests for greater and greater choice of lattice element type and configuration has led to ICOOL input decks becoming rather difficult to compose and modify easily. Moreover, the lack of a standard graphical post-processor has prevented many ICOOL users from extracting all but the most simple results from the output files. Here I present two attempts to improve this situation: First, a simple but quite general graphical pre-processor (NIME) written in the Tcl/TK[3] to permit users to write and maintain ASCII-formatted input files by use of simple macro definitions and expansions. Second, an interactive post-processor written in Fortran90 and NCAR graphics, which allows users to define, extract, and then examine the behavior of various particle subsets. In this paper I show some examples of use of both the pre- and post-processor for a standard ICOOL run.

1 INTRODUCTION

Given the exponential growth in computer processing speed, memory, and storage capability, there are numerous simulation codes in daily use in the accelerator physics community that serve as the basic workhorses of modern accelerator design. However, no matter how advanced and intelligent the physics package within a particular code, it can often be a frustrating challenge for the user to construct the necessary input files to achieve the wanted simulation parameters and/or, after a (hopefully) successful simulation run, extract detailed physics results from the often voluminous output files. The ICOOL simulation code[1], which was written to model the front end of a possible neutrino factory/muon collider facility, is one such code. ICOOL is in heavy use by mutiple users at several different laboratories but also requires a great deal of effort to construct proper input files. Moreover, there is no standard post-processor tool.

Following some stimulating conversations with R. Palmer(BNL) and G. Penn(UCB/LBNL) concerning the obstacles associated with ICOOL input files, I wrote a simple but very general, graphical pre-processor named "NIME" (for NIfty Macro Expander) to assist users in writing and maintaining ASCII-formatted input files by use of macro definitions and expansions. Section 2 gives details concerning the features and implementation of NIME. A second, more extensive, and continuing project was to write an interactive post-processor for ICOOL output. For various reasons (including multiplatform compatibility), the ICOOL code itself produces no direct graphical output but, instead, a number of ASCII-formatted output files. Specifically, these include the so-called "FOR009.DAT" file consisting of macroparticle "dumps" at discrete, user-specified, locations in z and the "FOR002.DAT" file which, among other things, archives details of macroparticle loss and decay. Together, these files contain an essentially complete and highly detailed record of the full phase space evolution of the transported muon beam. Section 3 discusses the philosophy and structure of this interactive post-processor which allows the user to define, extract, and then examine the behavior of various particle subsets.

2 THE NIME PRE-PROCESSOR

Underlying NIME is a relatively simple Tcl/TK[3] script which accepts a standard ASCII-formatted file as input and then, after scanning this file for various macro definitions, expands any macros in text to produce a final output file. Tcl/TK was chosen (*e.g.* over Python) partially due to author familiarity, partially due to Tcl's text string orientation (there is minuscule need for numerical evaluation), and mainly due to the richness and simplicity of the TK toolbox for GUI generation. The basic goal of NIME was to allow a user to generate multiple blocks of text from repeated use of a relatively few number of heavily commented macros. This scheme is particularly applicable to ICOOL input files because a large portion of the text lines which determine the transport lattice are repeated over and over again. Moreover, ICOOL input files permit essentially no comment annotation, which can make them extremely difficult to understand months later after their original generation. NIME, while "targeted" toward the generation and maintenance of ICOOL input decks, will actually work upon any ASCII file. Thus, NIME could prove useful, to give an example, for generating a series of input files for batch runs of other codes in which only a few lines or variables might change from run to run.

The structure of a NIME input file is quite simple. Comments may occur anywhere on a line and always begin with an exclamation point, as in Fortran90. Macro definitions always begin on a new line and may be placed *anywhere* in the file; *i.e.* usage of a particular macro may occur before its definition, thus allowing the user to group macro definitions together at the end of the file if so desired. Macros

* Work supported by the U.S. Department of Energy under Contract No. DE-AC03-76SF00098
† Email: fawley@lbl.gov

have unique, user-chosen names which aids in readability. Importantly, macros may also contain a number of internal "variables" whose names are flagged by a preceding percent symbol and whose values upon expansion may either be a default value or one optionally set by the user on the macro call line. Each macro may be called multiple times but, at present, macros may not call other macros (*i.e.* no nesting). A detailed manual, sample input file, and actual Tclk/Tk script are freely available on the Muon Collider Collaboration Web site[4]. The following short excerpt from an actual NIME input file for ICOOL shows some examples of macro definition and use:

```
\def block
SREGION
%length 1 %stepsize
1 0 %rmax(1.0)
NONE
0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
%material(VAC)
CBLOCK
0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
\enddef

\def rfcell
SREGION
%length  1  %stepsize
1  0.  %rmax
ACCEL
2. %freq %grad %phase(0.) 0. 0. 0. 0. 0.
VAC
NONE
0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
\enddef

!..the following is a 4-cell RF section
REPEAT
4               !--------- 4*201 MHz RF
\rfcell %length=0.3728 %stepsize=5.e-3 //
      %rmax=0.65 %freq=201.25 %grad=6.4
\block %length=100.e-6  %stepsize=10.e-6 //
   %rmax=0.25 %material=BE ! RF window
ENDR
! 2nd free drift space
\block %length=0.443 %stepsize=0.005 %rmax=0.5
```

While NIME or an equivalent is not exactly hard-core rocket science, anyone who has ever tried to put together a 50+ region ICOOL lattice from scratch should be able to appreciate the possible time-savings and error avoidance via macro definition and use.

3 AN INTERACTIVE GRAPHICAL POST-PROCESSOR FOR ICOOL

As explained in the Introduction, ICOOL is somewhat unusual amongst PIC codes in its use of particle dump files as a primary output diagnostic. Moreover, since each particle and its decay products are given a unique identifier within these dump files, multiple subsets of particles can be defined (and possibly color-coded), and then either plotted

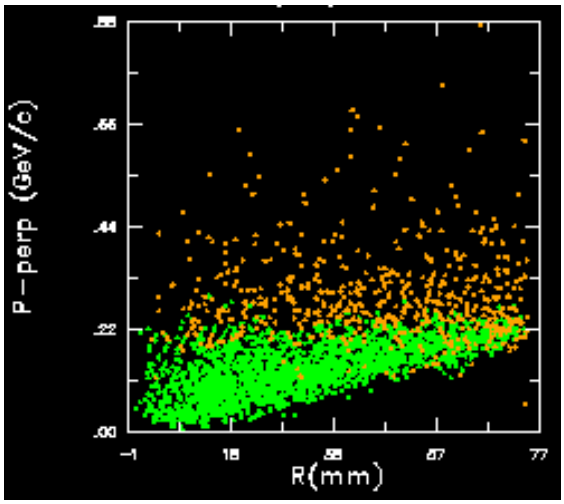

Figure 1: p_r versus r projection at $z = 0$ of macroparticles that survive to $z = 35$ m (green dots) and those which are lost (orange dots).

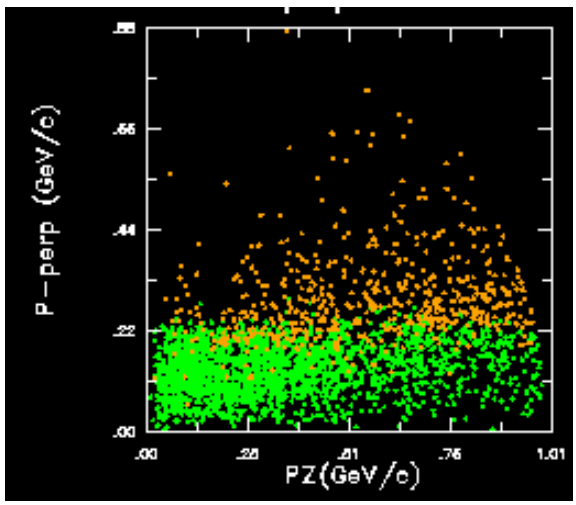

Figure 2: p_r versus p_z projection at $z = 0$ of particles that survive to $z = 35$ m (green dots) and those which are lost (orange dots).

directly in scatterplots or have average properties binned at any output location in the simulation. This then permits their transport properties to be then examined at both upstream and downstream locations. Example subsets might include those lost over particular intervals in z, those whose radii (or energy or transverse momentum and so on...) fall within given intervals at $z = 0$ (or any downstream z), *etc.* Perhaps more interestingly, if one sets up a subset to be those particles that end up within a particular 6-D phase space volume at some downstream position, one can determine various upstream "envelope" properties as a function of z and, perhaps, use this information to further optimize the transport lattice. Since ICOOL at present does not include collective forces (*e.g.* space charge), each macroparticle can be considered a test particle in 6D phase space.

The postprocessor (which presently exists without any

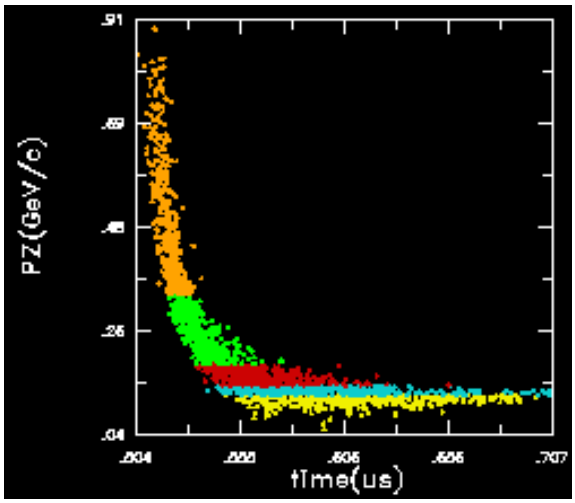

Figure 3: p_z versus time projection at $z = 152\,\mathrm{m}$ of macroparticles color coded by p_z value at the same z.

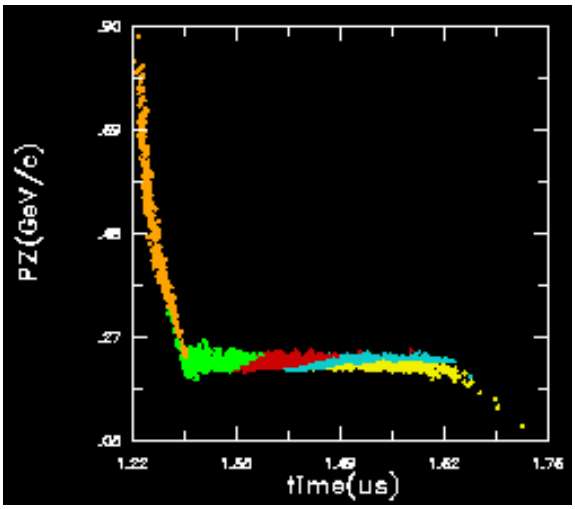

Figure 4: p_z versus time projection at $z = 364\,\mathrm{m}$ of surviving particles, with the same color coding as Fig. 3.b

snappy or clever name) is written in Fortran90 and relies upon NCAR graphics. I chose Fortran partly because ICOOL itself is a Fortran code and thus most of its users are Fortran proficient at some level. I chose F90 because the richness of its array syntax (including pointers) and its "TYPE" construct (similar to C structures) permits quite high level objects (albeit with limited inheritance capabilities when compared to C++) to be defined and used effectively. NCAR was a less certain choice and picked mainly because (1) it is essentially free and available for most UNIX/Linux platforms and (2) because I already had an extensive library of graphics routines built upon a NCAR/GKS foundation. Perhaps more importantly, I structured the postprocessor to use a command line driven interface (CLI) as its base default. While on one hand CLI's might be scoffed at as a fossil relic of the 1960's, they can be adapted both easily and *extremely rapidly* (with minimal changes to the underlying coding) to be driven by quite slick Expect[5]/TK GUI's with lots of fancy buttons, sliders, drop-down menus, etc.

As an example of the postprocessor's capabilities, I consider one of the relatively recent "non-distorting phase rotation" lattices [6] created by R. Palmer of BNL. This particular lattice ["STEP8" entitled *nd phase rotation with all matches graded (2.08 ndm8)*] begins at the proton target, followed by two induction accelerators and liquid H_2 minicool absorber sections for phase rotation, and finishes with $\sim 200\,\mathrm{m}$ of RF accelerator cells and cooling sections. Fig. 1 shows a post-processor produced scatterplot of the (p_r, r) transverse phase space projection at $z = 0$ color-coded by whether the particles survive to $z = 35\,\mathrm{m}$ or not. As is obvious, very few particles above an initial $p_r \geq 0.21\,\mathrm{GeV/c}$ survive and there is some additional loss at lower values of p_r for $r \geq 60\,\mathrm{mm}$. The equivalent (p_r, p_z) projection (Fig. 2) shows that this cutoff is essentially p_z (and thus energy) independent.

Figures 3 and 4 show the possible value of color coding particle subsets. The longitudinal phase space of in Fig. 3 is that predicted by ICOOL just beyond the first induction linac, which has succeeded in flattening out the $p_z - t$ curve for p_z values below $\approx 0.10\,\mathrm{GeV/c}$. The color coding corresponds to p_z values evaluated at $z = 152\,\mathrm{m}$ with equal macroparticle numbers per color group. Figure 4 shows the equivalent for the surviving particles at $z = 364\,\mathrm{m}$, just beyond the second and final induction linac, with the same individual macroparticle color coding. One sees that nearly all macroparticles whose original p_z at $z = 152\,\mathrm{m}$ fell below $\approx 0.34\,\mathrm{GeV/c}$ have been rotated by $z = 364\,\mathrm{m}$ to a relatively narrow band in p_z.

Quantitatively, macroparticle scatterplots become less than optimum when either the macroparticle number is large and/or many binning intervals(=# colors) are needed. An alternative in the postprocessor allows arbitrarily large bin number over one variable at a given z followed by historgram-like plots of some other property (averaged over the macroparticles in each bin) evaluated at one or more positions in z.

I am pleased to acknowledge useful discussions with G. Penn, A. Sessler, J. Wurtele, R. Fernow, and J. Gallardo. Computational resources at the DOE NERSC were utilized in this study.

4 REFERENCES

[1] Primary author R. Fernow, BNL; see http://pubweb.bnl.gov/people/fernow/icool/readme.htm

[2] A. M. Sessler, "Neutrino Factories: The Facility", http://www-mucool.fnal.gov/mcnotes/muc0155.pdf

[3] http://dev.scriptics.com/scripting/

[4] http://www.cap.bnl.gov/mumu/software/nime.html

[5] D. Libes, *Exploring Expect*, O'Reilly, Sebastopol, CA (1995).

[6] http://www-mucool.fnal.gov/mcnotes/muc0114.ps